Linux Grundlagen

Workshop am 21.09.2002

Torsten Höfler

Niederwürschnitz, den 15. Oktober 2002

1 Allgemeine Unix System V Konzepte

1.1 Bootvorgang (speziell SuSE Linux)

Folgeden Schrittfolge

- BIOS startet angegebenen Bootloader (aus dem Bootsektor des eingestellten Laufwerkes - Floppy, Festplatte, CDROM ...)
- im Allgemeinen wird LILO verwendet
- Initialisierungsschritte des LILO (Fehlerausgaben):

  L <errorcode> Der erste Teil des Bootloaders wurde geladen (vom Bootsektor des Laufwerkes), der zweite konnte nicht ausgeführt werden

  LI Der zweite Teil des Bootloaders wurde geladen (vom Laufwerk selbst), konnte aber nicht ausgeführt werden (Geometrieproblem)

  LIL Der zweite Teil des Bootloaders kann die Map-Table nicht nachladen (typischerweise Geometrie- oder Laufwerksfehler)

  LIL? Der zweite Teil des Boot Loaders wurde an eine falsche Adresse geladen

  LIL- Die Map-table des Boot-Loaders ist fehlerhaft

  LILO Bootloader ist komplett geladen

- der Bootloader lädt den gewählten Kernel (der Kernel kann auch direkt von BIOS geladen werden, darf dann aber nicht größer als 1MB sein)
- der Kernel entpackt sich selbst (falls gepackt), und führt seine Initialisierungsroutinen aus
- der Kernel checkt auf alle fest einkompilierten Hardwareoptionen
- der Kernel mountet die als root angegebene Partition (entweder im lilo, oder per Parameter root=<device>
- wenn diese Initialisierung abgeschlossen ist führt der Kernel /sbin/init aus
- der Init_prozess ist die Mutter aller anderen Prozesse
- init liest das Konfigurationsfile /etc/inittab (dort sind wichtige Optionen, wie standard runlevel, bootscript, die scripte für runlevel-Wechsel, die Standard Konsolen und gewisse Tastenkombinationen festgelegt)
- bei SuSE wird dann standardmäßig /etc/init.d/boot ausgeführt:
  - /proc wird gemountet
  - kerneld (autoloader) wird gestartet
  - evtl. Raid oder Lvm Volumes werden initialisiert
  - initialisieren der swap-Laufwerke
  - Laufwerke werden auf Filesystemfehler geprüft (kann deaktiviert werden, wenn das File /fastboot exisitert, und erzwungen werden, wenn das file /forcefsck existiert)
  - Modul Abhängigkeiten werden neu berechnet (depmod)
  - alle Filesysteme, die in der fstab auf auto stehen werden gemountet
  - Bibliothek-Abhängigkeiten werden neu berechnet (ldconfig)
  - loopback netzwerk device wird gestartet
  - hostname und domainname werden gesetzt

  - CMOS Uhr wird abgeglichen
  - Scripte in /etc/init.d/boot.d/S* werden gestartet
  - PNP wird initialisiert (isapnp)
  - /etc/init.d/boot.local wird ausgeführt

- danach geht der init in das eingestellte default runlevel, und führt das entsprechende script aus (/etc/init.d/rc <runlevel>)
- alle /etc/init.d/rc<alter-runlevel>.d/K* Scripte werden ausgeführt
- alle /etc/init.d/rc<neuer-runlevel>.d/S* Scripte werden ausgeführt

1.2 Init Scripte

- Init Scripte werden beim Bootvorgang oder beim Runlevel-Wechsel wie in Punkt **??** beschrieben gestartet
- SuSE < 8.0: die meisten Scripte werden in der rc.config konfiguriert (Starten kann duchr Variablen in rc.config verhindert werden)
- SuSE > 8.0: Scripte werden in /etc/conf.d/<script> konfiguriert (das Starten kann nur durch Entfernen des Links im Runlevel Verzeichnis verhindert werden - entpricht allgemeiner Linux Norm)

```
#!/bin/sh
##################################################################

# Importieren aller Variablen/Einstellungen
. /etc/rc.status
. /etc/rc.config

# falls direkt von der Konsole aus aufgerufen
base=${0##*/}
link=${base#*[SK][0-9][0-9]}

test $link = $base && CRON=yes

# fuehre aus, oder verlasse script
test "$CRON" = yes || exit 0

rc_reset
case "$1" in
    start)
        echo -n "Starting <my> daemon"
    startproc <path-to-daemon>
    rc_status -v
    ;;
     stop)
    echo -n "Shutting down <my> daemon"
    killproc -TERM <path-to-daemon>
    rc_status -v
    ;;
     restart)
    $0 stop && $0 start
    rc_status
    ;;
```

```
esac
rc_exit

################################################################################
```

1.3 Runlevel Konzept

- der Runlevel, in den gebootet wird kann dem Kernel übergeben werden (im LILO z.B. LILO: linux 1 - bootet in Runlevel 1, sehr nützlich bei Netzwerkproblemen)
- Runlevel wechsel durch: init <neuer-runlevel>
- Runlevel 0: System anhalten
- Runlevel 1: Single User Mode: Nur ein Benutzer kann arbeiten, meistens root. Es sollten nur die wichtigsten Dienste gestartet sein
- Runlevel 2: Mulituser, no network: Es können mehrere Benutzer arbeiten, ohne Netzwerk-Exports (NFS) (multiuser with no network services exported)
- Runlevel 3: Normal, Multiuser: Normaler Modus
- Runlevel 4: Reserviert: normale Benutzung, Multiuser
- Runlevel 5: Multiuser mit X-Anmeldung: Es erscheint der X-Server zur Benutzer-Anmeldung
- Runlevel 6: Reboot: Rechner wird neugestartet

2 Konzepte der Bash

2.1 Einfache Bash-Scripte

```
#!/bin/bash

echo "Hello World"
```

2.2 Output Redirection

Grundlegend gibt es drei Standard Streams: stdin, stdout, stderr (std=standard).
Weiterleitungsmöglichkeiten:

stdout nach datei z.B. ls > ls.txt

stderr nach datei z.B. ls 2> ls-err.txt

stdout nach stderr z.B. ls 1>&2

stderr nach stdout z.B. ls 2>&1

stderr und stdout nach datei z.B. ls &> ls-all.txt

2.3 pipes

Mittels pipes kann man die Ausgaben eines Programmes als Eingaben einens zweiten Programmes nutzen (beliebig oft wiederholbar).

```
z.B. ls -1 | grep test
```

2.4 Variablen

Bash Variablen entsprechen allgemeinen Variablen, mit den Ausnahmen, dass sie nicht deklariert werden müssen und keinen Datentyp besitzen.
Verwendung:

```
<Variablennane>=<Inhalt>
z.B. TEST="HALLO"
```

Mit Variablen kann man auch rechnen:

```
X=2
# X=2
echo $X
# 1. Variante:
let X=X+5
# X=7
echo $X
# kurze Variante:
X=$[$X+5]
# X=12
echo $X
```

und Werte einlesen:

```
read X
echo $X;
```

Programmrückgaben in Variablen schreiben oft ist es sinnvoll, die Rückgabewerte eines Programms in eine Variable zu schreiben, dies geschieht folgendermassen:

```
DATE=$(date)
```

in DATE steht jetzt der Rückgabewert von Kommando DATE (wenn dieser mehrzeilig ist werden die Zeilenumbrüche entfernt!)

```
echo $DATE
```

2.5 Bedingungen

Mittels Bedingungen kann man entscheiden, eine Aktion auszuführen oder nicht.
Wird mittels eines if then else Konstruktes realisiert.
Der Syntax lautet:

```
if [bedingung];
then
  $<$code$>$
else
  $<$code$>$
fi;
```

z.B.

```
A="bla"
B="blub"

if [ "$A" = "$B" ];
```

```
then
    echo "$A = $B"
else
    echo "$A != $B"
fi;
```

2.6 Schleifen

2.6.1 for-Schleifen

Die for-Schleife unterscheidet sich stark von anderen Programmiersprachen, sie entspricht eher der foreach Schleife. Mit ihr kann man eine Reihe von Wörtern durchgehen. Syntax:

```
for i in a b c d e;
do
    echo $i;
done;
```

ebenso kann man wieder den Output eines Programmes nehmen, dabei gelten Leerzeichen und Zeilenumbrüche als „Trennzeichen“.
z.B. alle Dateien in einem Verzeichnis ausgeben.

```
for i in $(ls -1);
do
    echo $i;
done;
```

oder for wie in C:

```
for i in $(seq 1 10);
do
    echo $i;
done;
```

2.6.2 while-Schleifen

Schleife wiederholen solange eine Bedingung erfüllt ist.
z.B.

```
i=0;
while [ $i -lt 10 ];
do
    echo $i;
    let i+=1;
done;
```

2.6.3 until-Schleifen

Schleife abbrechen, wenn Bedingung erfüllt.
z.B.

```
i=0;
until [ $i -gt 10 ];
do
    echo $i;
    let i+=1;
done;
```

2.7 Funktionen

Funktionen wie aus anderen Programmiersprachen bekannt.

```
function hello()
{
   echo "hello";
}

# Funktion mit Parametern vgl. Perl
function say()
{
   echo $1;
}

hello;
say "test";
```

2.8 einfache Userinterfaces

Mittels des select-Befehls.

```
OPT="Hello Quit"
select opt in $OPT;
do
   if [ "$opt" = "Quit" ];
   then
      echo "quit"
   elif  [ "$opt" = "Hello" ];
   then
      echo "hello"
   else
      echo "bad option"
   fi
done
```

2.9 Sinnvolle Tools

sed Unix Stream Editor - zum Bearbeiten von Zeilen eines Streams

awk Änderungen an Dateien / Streams

tput Gibt Terminal-Infos aus

cat liest eine Datei und gibt sie nach stdout

tac das gleiche wie cat, nur von hinten beginnend

less Anzeige eines Files Streams Seitenweise

more siehe less

grep zeigt nur die Zeilen, die einem Muster entsprechen

sort Sortiert Zeilen eines Files

wc zählt Zeichen, Wörter und Zeilen einer Datei / eines Streams

... und viele mehr.

2.10 bash debugging

Falls mal etwas zu debuggen ist, einfach im Header
`#!/bin/bash -x`
eintragen.

3 Serverdienste

3.1 Sendmail

3.1.1 das smtp-Protokoll

normaler Ablauf z.B. mail an htor@mail.unixer.de:

- Sendmail trennt domainteil aus der Adresse -> mail.unixer.de
- Sendmail sucht zuständigen Mailserver für die domain (den MX-Nameserver Record)
- Sendmail baut zu dem Remote-Server eine smtp-Verbindung auf
- der Remote-Server sendet eine Begrüssungsnachricht (mit Status 220)
- Sendmail sagt helo (helo mail.unixer.de)
- Remote-Server antwortet mit Status 250 Nachricht
- Sendmail sendet Absendermailadresse (mail from: htor@unixer.de)
- Remote-Server antwortet mit Status 250
- Sendmail sendet Empfänger-Mailadresse (rcpt to: htor@mail.unixer.de)
- Remote-Server antwortet mit Status 250
- Sendmail sendet data-Kommando (data)
- Remote Server antwortet mit Status 254
- Sendmail sendet body der Mail abgeschlossen mit „.“ allein auf einer Zeile
- Remote Server sendet Status 250
- Sendmail beendet Verbindung (quit)
- Remote-Server sendet Status 221 und schliesst TCP-Verbindung

3.1.2 Sendmail Konfiguration

Die sendmail Konfiguration ist komplett in der Datei /etc/sendmail.cf und im Verzeichnis /etc/mail vorzunehmen. Die Datei /etc/sendmail.cf ist ausreichend kommentiert.

3.1.3 alias Definitionen

In der Datei /etc/aliases kann man Aliase für lokale Mailadressen festlegen (ist im Header kommentiert). Nachdem das file angepasst wurde muss das Programm newaliases gestartet werden, damit die Änderungen wirksam werden.

3.2 POP-Server

3.2.1 das POP-Protokoll

normaler Ablauf einer POP-Sitzung:

- Mailclient baut Verbindung zum POP-Server auf
- Pop-Server antwortet mit Begrüssungsmessage
- Mailclient sendet user-Kommando (user <username>)
- Pop-Server sendet +Ok Status
- Mailclient sendet pass-Kommando (pass <passwort>)
- Pop-Server sendet +Ok Status mit der Anzahl der Mails und der Gesamtgrösse
- weitere Kommandos:
  - list - zeigt eine Liste aller Messages und deren Grösse an
  - retr <nr> - fordert Nachricht <nr> an
  - dele <nr> - löscht Nachricht <nr>
  - quit - schliesst die Verbindung zum Pop-Server

3.2.2 Konfiguration

Der Pop-Server (popper) kann kaum konfiguriert werden. Eine Zugriffsbeschränkung ist allgemein über den tcpd möglich. Der popper wird gewöhnlicherweise vom inetd gestartet, und wird auch in der /etc/inetd.conf konfiguriert.

3.3 HTTP-Server

3.3.1 HTTP-Protokoll

- Client baut Verbindung zum HTTP-Server auf und sendet den Request.
- der Request ist folgendermassen aufgebaut:
  GET /<file> HTTP/1.0
  <Header>
  <zwei Leerzeilen>

Header müssen folgendermassen aussehen: <name>: <wert>

3.3.2 Konfiguration

Die Konfiguration des Apache Webservers findet in der Datei /etc/httpd/httpd.conf statt. Diese Datei ist ausreichend kommentiert. Einstellungen am PHP können über die Datei /etc/httpd/php.ini vorgenommen werden.